	b)	Describe the three types of scripting elements in JSP with examples.	L2	CO3	5 M
		OR			
9	a)	Illustrate difference between static include (<%@ include %>) and dynamic include (<jsp: include>) with code snippets.	L3	CO3	5 M
	b)	Prepare a sample JSP page using <c: forEach> and <c: if> tags from JSTL and explain their purpose.	L3	CO3	5 M
		UNIT-V			
10	a)	Draw and explain the architecture of the Spring Framework. Briefly describe the role of any four major modules.	L2	CO4	5 M
	b)	Illustrate the difference between constructor-based and setter-based dependency injection in Spring with examples.	L3	CO4	5 M
		OR			
11	a)	How does setter injection work in Spring? Prepare a code example using XML configuration.	L2	CO4	5 M
	b)	Describe the Spring MVC life cycle with a neat diagram. Explain the role of DispatcherServlet.	L2	CO4	5 M

PVP 23

Code: 23IT3501

III B.Tech - I Semester - Regular Examinations - NOVEMBER 2025

ADVANCED JAVA
(INFORMATION TECHNOLOGY)

Duration: 3 hours Max. Marks: 70

Note: 1. This question paper contains two Parts A and B.
2. Part-A contains 10 short answer questions. Each Question carries 2 Marks.
3. Part-B contains 5 essay questions with an internal choice from each unit. Each Question carries 10 marks.
4. All parts of Question paper must be answered in one place.

BL – Blooms Level CO – Course Outcome

PART – A

		BL	CO
1.a)	Which JDBC driver type is platform-independent and why?	L2	CO2
1.b)	Explain ACID properties in the context of JDBC transactions.	L2	CO2
1.c)	Differentiate between monolithic and multi-tier architecture.	L2	CO1
1.d)	Mention disadvantages of Model-I architecture.	L2	CO1
1.e)	How is a servlet different from a CGI program? Explain.	L2	CO3
1.f)	Explain the use of getSession() method in session tracking?	L2	CO3
1.g)	What is JSP? Explain advantage of JSP over Servlets.	L2	CO3
1.h)	Identify which JSP tags or features are commonly used in CRUD operations.	L2	CO3

1.i)	What is the Spring Framework? Identify its features.	L2	CO4
1.j)	Identify any two commonly used annotations in Spring configuration.	L2	CO4

PART – B

			BL	CO	Max. Marks
		UNIT-I			
2	a)	Explain the JDBC architecture with a neat diagram. How does it facilitate database connectivity in Java applications?	L2	CO2	5 M
	b)	What are the common methods in ResultSet interface? Explain how they are used in JDBC with an example.	L2	CO2	5 M
		OR			
3	a)	Explain the use of CallableStatement in JDBC. Construct a sample code to call a stored procedure.	L3	CO2	5 M
	b)	Interpret different types of transaction isolation levels in JDBC. How do they effect concurrency?	L3	CO2	5 M
		UNIT-II			
4	a)	Identify the difference between a web server and an application server. Give examples of each. Also explain different types of HTTP request methods.	L2	CO1	5 M
	b)	Describe the main components of a web application. How do servlets and JSPs interact?	L2	CO1	5 M
		OR			
5	a)	What are EE6 containers? Describe the types of containers provided by EE6 with examples.	L2	CO1	5 M
	b)	Describe the Model-2 (MVC) architecture with a neat diagram. How does it improve over Model-I?	L2	CO1	5 M
		UNIT-III			
6	a)	Describe the performance benefits of using servlets over other server-side technologies.	L2	CO3	5 M
	b)	Illustrate the use of Cookie class in session tracking with proper code snippet and also explain methods of Cookie class.	L3	CO3	5 M
		OR			
7	a)	Explain the directory structure of a servlet-based web application and the role of the web.xml file.	L2	CO3	5 M
	b)	How is the HttpServletResponse interface used to send responses? Explain setting headers and redirecting responses.	L2	CO3	5 M
		UNIT-IV			
8	a)	Explain the different phases in the JSP life cycle. Describe the role of methods like jspInit(), _jspService(), and jspDestroy().	L2	CO3	5 M

PVP23

Code: 23IT3501

III B.Tech - I Semester – Regular Examinations
November 2025
ADVANCED JAVA
(INFORMATION TECHNOLOGY)

Duration: 3 hours Max. Marks: 70

Note: 1.This question paper contains two Parts A and B.
2. Part –A contains 10 short answer questions. Each Question carries 2 Marks.
3. Part-B contains 5 essay questions with an internal choice from each unit. Each Question Carries 10 marks
4. All parts of Questions must be answered in one place.
BL- Blooms Level CO-Course Outcome

PART – A

1 .a)**Which JDBC driver type is platform-independent and why.** 2M

Identification of JDBC driver type and why →2M

1.b)**Explain ACID properties in the context of JDBC transactions.** 2M
Explanation of ACID properties → 2M

1.c)**Differentiate between monolithic and multi-tier architecture.** 2M
Any two differences between monolithic and multi-tier architecture -→ 2M

1.d)**Mention disadvantages of Model-I architecture** 2M
Any two disadvantages of Model-I architecture→ 2M

1.e)**How is a servlet different from a CGI program?Explain.** 2M

Any two differences of servlet and CGI program →2M

1.f)**Explain the use of getSessionO method in session tracking?** **2M**
uses of getSessionO method -→ 2M

1.g)**What is JSP? Explain advantage of JSP over Servlets ?** **2M**
What is JSP → 1M
Advantage of JSP over Servlets -→1M

1.h)**IdentifywhichJSP tagsorfeaturesarecommonly used in CRUD operations.2M**
Identification of JSP tags used in CRUD operations → 2M

1.i)**What is the Spring Framework? Identify its features.** **2M**

What is spring frame work -→1M
Any two features →1M

j)**Identify any two commonly used annotations in Spring configuration. 2M**
Any two used annotations in Spring configuration → 2M

18112523B -47

PART - B
UNIT-I

2. a. **Explain the JDBC architecture with a neat diagram. How does it facilitate database connectivity in Java applications?.5 M**

Explanation of JDBC Architecture →2M
Diagram → 1M
database connectivity in Java applications→2M

b)**What are the common methods in ResultSet interface? Explain how they are used in JDBC with an example.5 M**

Any 3 common methods in ResultSet interface →3M
Explanation of how they are used in JDBC with an example→2M

OR

3. a)**Explain the use of CallableStatement in JDBC. Construct a sample code to call a stored procedure. 5M**

Explanation of CallableStatement in JDBC→ 3M
sample code to call a stored procedure → 2M

b**Interpret different types of transaction isolation levels in JDBC. How do they effect concurrency. 5M**

Explanation of different types of transaction isolation levels in JDBC→ 3M
What is the effect of concurrency → 2M

UNIT-II

4a)**Identify the difference between a web server and an application server. Give examples of each. Also explain different types of HTTP request methods. -- 5M**

Difference between a web server and an application server with an example →3M
Any two types of HTTP request methods →2M

b)**Describe the main components of a web application. How do servlets and JSPs interact?5M**

What are the main components of a web application? → 3M
How servlets and JSPs interact →2M

OR

5a)**What are EE6 containers? Describe the types of containers provided by EE6 with examples.5 M**

What are EE6 containers→ 2M
Explanation of types of containers provided by EE6 with examples→ 3M

b) Describe the Model-2 (MVC) architecture with a neat diagram. How does it improve over Model-I?. 5M

What is MVC architecture → 2M
Diagram →1M

How does it improve over Model-I →2M

UNIT-III

6.a)**Describe the performance benefits of using servlets over other server-side technologies.5M**

Any five benefits of using servlets over other server-side technologies →5 M

b)**Illustrate the use of Cookie class insession tracking with proper code snippet and also explain methods of Cookie class 5M**

what is the use of Cookie class in session tracking →1M
Relevant example Program → 2M
methods of Cookie class → 2M

OR

7.a)**Explain the directory structure of a servlet-based web application and the role of the web.xml file 5M**

Explanation of directory structure of a servlet-based web application → 4M
Role of web.xml→ 1M

b) **How is the HttpServletResponse interfaceused to send responses? Explain setting headers and redirecting responses. 5M**

How the HttpServletResponse interface used to send responses→**3M**
Explanation of setting headers and redirecting responses.→**2M**

UNIT-IV

8a)**Explain the different phases in the JSP life cycle. Describe the role of methods like jsplnit(), jspServiceQ, and jspDestroy(). 5M**

Explanation of different phases in the JSP life cycle→**3M**
Description of **jsplnit(),jspServiceQ and jspDestroy().→ 2M**

b) **Describe the three types of scripting elements in JSP with examples.-5M**

Explanation of three types of scripting elements in JSP with examples→ 5M

OR

9a)**Illustratedifferencebetween static include (<°lo@ include %>) and dynamic include (<jsp: include>) with code snippets.**--5M

Difference between static include and dynamic include →3M
Example programs → 2M

b) **Prepare a sample JSP pageusing<c: forEach> and <c: if.> tags from JSTL and explain their purpose. --5M**

sample JSP Program using <c: forEach> and <c: if.> tags from JSTL→ 3M
purpose of each tag → 2M

UNIT-V

10a)**Draw and explain the architecture of theSpring Framework. Briefly describe the role of any four major modules.5 M**

Explanation of architecture of spring and major modules →3M

Diagram of Spring Framework → 2M

b)**Illustratethe difference between constructor-based and setter-based dependency injection in Spring with examples.**

Any two differences constructor-based and setter-based dependency injection→ 3M

Examples of each →**2M**

OR

11a)**How does setter injectionworkin Spring? Prepare a code example using XML configuration.5**M

Explanation of how setter injection work in Spring →3M

Relevant example program → 2M

b)**Describe the Spring MVC life cycle with a neat diagram. Explain the role of DispatcherServlet**5 M

Explanation of Spring MVC life cycle→ 2M

Diagram → 1M

Role ofDispatcherServlet→ 2M

PVP23

Code: 23IT3501

IIIB.Tech - I Semester – Regular Examinations
November 2025
ADVANCED JAVA
(INFORMATION TECHNOLOGY)

Duration: 3 hours Max. Marks: 70

Note: 1.This question paper contains two Parts A and B.
2. Part –A contains 10 short answer questions. Each Question carries 2 Marks.
3. Part-B contains 5 essay questions with an internal choice from each unit. Each Question
Carries 10 marks
4. All parts of Questions must be answered in one place.
BL- Blooms Level CO-Course Outcome

PART – A

1 .a)**Which JDBC driver type is platform-independent and why.** 2M

The **Type 4 JDBC driver (Thin Driver)** is platform-independent because it is written entirely in **Java** and directly communicates with the database using its native protocol. It does not need any native libraries or ODBC bridge, so it can run on any system that supports Java.

1.b)**Explain ACID properties in the context of JDBC transactions.** 2M

ACID properties ensure reliability and consistency of transactions in JDBC.
- **Atomicity:** All operations in a transaction complete successfully, or none are applied.
- **Consistency:** The database remains in a consistent state before and after the transaction.
- **Isolation:** Each transaction is independent and does not interfere with others.
- **Durability:** Once a transaction is committed, changes are permanent even after a system crash.

1.c)**Differentiate between monolithic and multi-tier architecture.** 2M

Monolithic Architecture	**Multi-tier Architecture**
All components (UI, logic, data) are combined in one single unit.	Application is divided into layers like Presentation, Business, and Data.
Difficult to modify, scale, or test.	Easier to maintain, update, and scale.
Tight coupling between parts.	Loose coupling, modular design.
Example: JSP with JDBC code in same file.	Example: JSP (UI) → Servlet (Logic) → DAO (Database).

1.d)**Mention disadvantages of Model-I architecture** 2M

1. Business logic and presentation logic are mixed together in JSP files.
2. Difficult to maintain and test large applications.
3. Low reusability of code.
4. Poor separation of concerns.
5. Debugging and scalability become harder.

1.e)**How is a servlet different from a CGI program?Explain.** 2M

Servlet	**CGI(Common Gateway Interface)**
• Servlets are portable and efficient. • In Servlets, sharing data is possible. • Servlets can directly communicate with the webserver. • Servlets are less expensive than CGI • Servlets can handle the cookies	• CGI is not portable • In CGI, sharing data is not possible. • CGI applications cannot directly communicate with the webserver(Some more interface is required). • CGI is more expensive than Servlets • CGI cannot handle the cookies.

1.f)**Explain the use of getSessionO method in session tracking?** **2M**

The getSession() method of the HttpServletRequest object is used to **retrieve or create a user session**.

- getSession(true) → returns current session or creates a new one if not found.
- getSession(false) → returns existing session or null if not found.

1.g)**What is JSP? Explain advantage of JSP over Servlets ?** **2M**

JSP (JavaServer Pages) is a server-side technology that combines HTML with Java code to create dynamic web pages.

Advantages over Servlets:

1. Easier to write and maintain as most of the code is HTML.
2. Automatic translation into Servlets by the server.
3. Better separation of presentation and business logic.
4. Supports Expression Language and Tag Libraries.

1.h)**IdentifywhichJSPtagsorfeaturesarecommonly used in CRUD operations.2M**

Commonly used JSP features for CRUD operations are:

- **JSTL SQL Tags:** `<sql:query>`, `<sql:update>`for database operations.
- **Core Tag:** `<c:forEach>`to iterate over records.
- **Expression Language (EL):** `${}` syntax for accessing data.
- **Scriptlets (`<% %>`):** For embedding Java code (less preferred).

1.i)**What is the Spring Framework? Identify its features.** **2M**

The **Spring Framework** is a lightweight, open-source framework for building enterprise-level Java applications.

Features:

1. **Inversion of Control (IoC)** and **Dependency Injection (DI)**.
2. **Aspect-Oriented Programming (AOP)** support.
3. **Spring MVC** for web applications.
4. **Transaction management** and integration with ORM tools like Hibernate.
5. **Modular architecture** and testability.

1. j)**Identify any two commonly used annotations in Spring configuration. 2M**

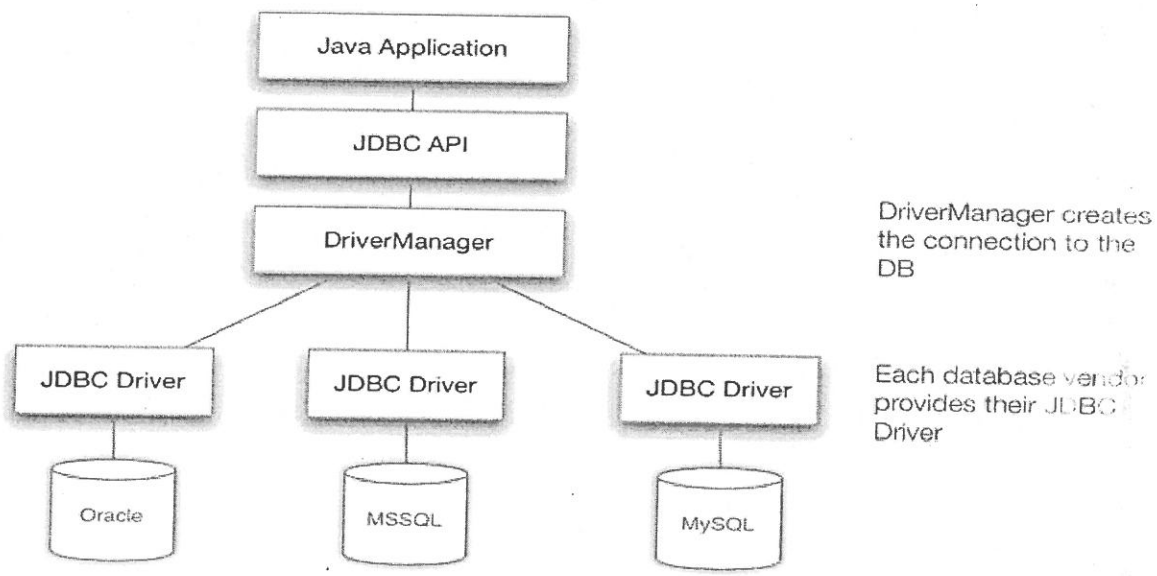
Autowired: Automatically injects dependencies into beans.
Component / @Controller / @Service / @Repository: Used to define Spring-managed beans.

PART - B
UNIT-I

2. a. **Explain the JDBC architecture with a neat diagram. How does it facilitate database connectivity in Java applications?.**5 M

JDBCArchitecture

- The JDBC API supports both two-tier and three-tier processing models for database access but in **general JDBC Architecture** consists of two layers:
  - **JDBCAPI:**Thisprovidestheapplication-to-JDBCManagerconnection.
  - **JDBCDriverAPI:**ThissupportstheJDBCManager-to-DriverConnection.
- The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.
- The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.
- Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application:



Common JDBC Components:

How it facilitates connectivity: Application loads JDBC driver, gets a Connection, creates Statement/PreparedStatement, executes SQL, reads results from ResultSet, and closes resources. The driver handles network/protocol conversion so application code is DB-independent.

```
Class.forName("com.mysql.cj.jdbc.Driver");
Connection conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/shop","user","pass");
PreparedStatementps = conn.prepareStatement("SELECT id,name FROM product WHERE price >
?");
```

```
ps.setDouble(1, 100.0);
ResultSetrs = ps.executeQuery();
while(rs.next()){
System.out.println(rs.getInt("id")+" "+rs.getString("name"));
}
rs.close(); ps.close(); conn.close();
```

b)What are the common methods in ResultSet interface? Explain how they are used in JDBC with an example.5 M

Common methods (with purpose):

- boolean next() — move cursor to next row; returns false when no more rows.
- getString(String/ int) — retrieve column value as String.
- getInt(String/ int) — retrieve column value as int.

getDouble(), getDate(), getTimestamp() — typed getters.

- booleanwasNull() — check if last read column was SQL NULL.
- close() — free resources.
- getMetaData() — obtain ResultSetMetaData.
- previous(), first(), last() — cursor positioning (only if scrollable).
- updateXXX() and updateRow() — for updatable ResultSets.

Usage example:

```
try {
      Connection conn = DriverManager.getConnection(url, username,
password);
      Statement stmt = conn.createStatement();
ResultSetrs = stmt.executeQuery("SELECT * FROM your_table");

      while (rs.next()) {
           // Process the result set
       }

rs.close();
stmt.close();
conn.close();
}
catch (SQLException e) {
e.printStackTrace(); // Handle the exception
}
```

OR

3. a)**Explain the use of CallableStatement in JDBC. Construct a sample code to call a stored procedure. 5M**

Explanation: CallableStatement is used to call database stored procedures/functions. Supports IN, OUT and INOUT parameters and can return ResultSets.

Sample stored procedure (SQL, e.g., MySQL):

```
DELIMITER //
CREATE PROCEDURE data100 (OUT var1 INT)
```

```
BEGIN
SELECT max(sal) INTO var1 FROM EMP;
END;
//Java code to call it:
importjava.util.*;
importjava.sql.*;
public class CallableStatementwithOut{
public static void main(String args[])
{
int number;
try{
Class.forName(&amp;quot;com.mysql.jdbc.Driver&amp;quot;);
Connection
con=DriverManager.getConnection(&amp;quot;jdbc:mysql://localhost:3306/sri?characterEncoding=utf8
&amp;quot;,&amp;quot;root&amp;quot;,&amp;quot;p
vpsit&amp;quot;);
CallableStatementcs=con.prepareCall(&amp;quot;{call data1001(?)}&amp;quot;);
cs.registerOutParameter(1,Types.INTEGER);
cs.execute();
System.out.println(&amp;quot;Maximum Salary:&amp;quot;+cs.getInt(1));
con.close();
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

3.b)**Interpret different types of transaction isolation levels in JDBC. How do they effect concurrency. 5M**

JDBC defines constants in Connection for isolation levels:

1. 1READ UNCOMMITTED:

   - Lowestlevelof isolation.
   - Allowstransactionsto**readuncommitted(dirty)data**fromother transactions.
   - Fastestbutleastsafe.

2. READ COMMITTED
   - DefaultlevelinmanyDBMS(e.g.,Oracle).
   - Preventsdirtyreads.
   - Eachread getsthe**latestcommittedvalue**.

3. REPEATABLEREAD
   - Ensures**samedata**isreturnedif queriedmultipletimeswithina transaction.
   - Preventsdirtyandnon-repeatablereads.
   - Stillallows**phantomreads**.

4. SERIALIZABLE

- Highestisolation level.
- Fullyisolates transactions — **likeexecuting themoneafteranother**.
- Slowerdueto lockingbutensures consistency.

- **setTransactionIsolation() and getTransactionIsolation( )-** These two methods are part of the java.sql.Connection interface and are used to manage transaction isolation levels in JDBC.
  Ex:
  connection.setTransactionIsolation(Connection.TR
  ANSACTION_SERIALIZABLE);

  intlevel= connection.getTransactionIsolation();

How used in JDBC:

```
conn.setAutoCommit(false);
conn.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);
try {
// SQL operations
conn.commit();
} catch(Exception e){
conn.rollback();
}
```

UNIT-II

4a)**Identify the difference between a web server and an application server. Give examples of each. Also explain different types of HTTP request methods. -- 5M**

Web server vs Application server (difference):

- **Web Server**
  o Serves static content (HTML, CSS, images).
  oHandles HTTP requests, basic CGI support.
  oExamples: **Apache** .

- **Application Server**
  oProvides runtime for server-side business logic (servlets, EJBs, JMS, JTA).
  o Supplies services: transaction management, security, connection pooling.
  oExamples: **Apache Tomcat** (lightweight servlet container —container), **GlassFish**, **WebLogic**, **WebSphere**.

HTTP request methods :

- **GET** — retrieve resource; safe & idempotent; parameters in URL.
- **POST** — submit data to server (create or process); not idempotent.
- **PUT** — replace or create resource; idempotent.
- **DELETE** — delete resource; idempotent.
- **HEAD** — like GET but only headers returned.
- **OPTIONS** — describe communication options for resource.
- **PATCH** — partial update of resource (not necessarily idempotent).
- **TRACE** — diagnostic; echoes request (rarely used due to security).

b)**Describe the main components of a web application. How do servlets and JSPs interact?5 M**

Main components:

- **Client (Browser)** — makes HTTP requests.
- **Web server / Servlet container** — receives requests, forwards to servlets/JSPs.
- **Servlets** — Java classes handling request/response logic.
- **JSPs** — page templates that generate HTML (compiled to servlets).
- **Business layer** — services, EJBs or POJOs handling business logic.
- **Data layer** — JDBC/ORM interacting with DB.
- **Deployment descriptor (web.xml)** or annotations — configure mappings, filters, listeners.

➢ **Servlets and JSP interaction:**

• JSPs are primarily presentation; they get compiled into servlet classes by the container.
• A common pattern: servlet acts as a controller (processes request, sets model attributes), then forwards to JSP for rendering:

OR

5a)**What are EE6 containers? Describe the types of containers provided by EE6 with examples.5 M**

Java EE 6 containers are runtime environments provided by an application server that manage lifecycle, services and resources for Java EE components.

Types of containers in Java EE 6:

1. **Web Container** (Servlet container)
   - Manages servlets, JSPs, filters, listeners.
   - Example: Tomcat (servlet/JSP), GlassFish web container within full server.
2. **EJB Container**
   - Manages EJB components (session beans, message-driven beans), lifecycle, transactions, security.
   - Example: JBoss/WildFly EJB container, GlassFish.
3. **Application Client Container**
   - Executes application client components (desktop Java clients) with Java EE services.
4. **Resource Adapter Container (Connector Container)**
   - Manages JCA resource adapters (connectivity to legacy systems).
5. **Web Services Runtime** (part of container features)
   - Supports JAX-WS / JAX-RS endpoints.

Example servers that implement EE6: GlassFish 3.x, JBoss AS 7 / WildFly (implements Java EE).

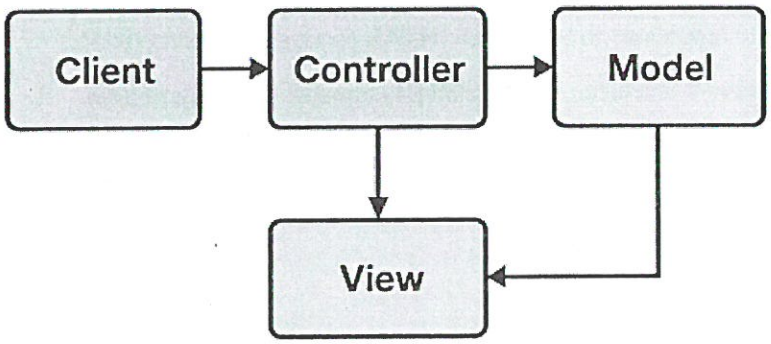
b) Describe the Model-2 (MVC) architecture with a neat diagram. How does it improve over Model-I?. 5M

Model-2 (MVC) explanation:

• **Model:** business/data logic (Java beans, services).

- **View:** JSPs — presentation templates.
- **Controller:** Servlets — receive requests, interact with model, select view.

Model-2 (MVC) Web Architecture



How it improves over Model-1 (JSP-centric):

- **Separation of concerns:** Controller handles control flow; JSP only for view. Model-1 mixes logic into JSPs, leading to maintainability issues.
- **Easier testing:** Business logic isolated in Java classes (Model).
- **Reusability:** Controllers can forward to multiple views; views simpler and reusable.
- **Scalability & maintainability:** Cleaner design for larger apps.

UNIT-III

6.a)**Describe the performance benefits of using servlets over other server-side technologies.5 M**

- **Single JVM, persistent objects:** Servlet instances are created once; doGet/doPost invoked on threads — avoids process creation per request (unlike CGI).
- **Threaded model:** Multiple requests handled by threads within same process, low overhead.
- **Compiled Java code:** Faster than interpreted scripts; JIT optimized.
- **Connection pooling & resource reuse:** Containers provide pools for DB connections, threads, reducing expensive operations.
- **Caching & filters:** Easy to implement caches, compression, filters at container level.
- **Efficient lifecycle:** init() called once; destroy() once; service() handles requests.

b)**Illustrate the use of Cookie class insession tracking with proper code snippet and also explain methods of Cookie class 5M**

Cookie usage (session tracking):

HTTP is a stateless protocol. It does not remember user interactions automatically.
Cookies are small pieces of text data stored by the browser, sent by the server, and used tomaintain state between requests.
In Java Servlets, the Cookie class (from javax.servlet.http.Cookie) is used to create, read, andmanage cookies for session tracking.
Cookies help identify users across multiple requests, enabling:

- User login persistence
- Shopping cart tracking

- Personalization
- Session identification

Important Cookie methods:

- Cookie(String name, String value) — constructor.
- setMaxAge(int seconds) — lifetime (0 delete, -1 session).
- setPath(String path) — URL path for which cookie sent.
- setDomain(String domain) — cookie domain.
- setSecure(boolean) — send only over HTTPS.
- setHttpOnly(boolean) — not accessible via client-side scripts (prevents XSS theft).
- getName(), getValue() — accessors.

a) Setting a Cookie on the Server (Servlet 1)

```
import java.io.*;
importjavax.servlet.*;
importjavax.servlet.http.*;
public class SetCookieServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
// Create a cookie
Cookie userCookie = new Cookie("username", "Anjali");
// Set cookie properties
userCookie.setMaxAge(60 * 60 * 24); // valid for 1 day
userCookie.setPath("/");
// Add cookie to response
response.addCookie(userCookie);
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("Cookie created and sent to client.");
}
}
```

b) Retrieving the Cookie (Servlet 2)

```
import java.io.*;
importjavax.servlet.*;
importjavax.servlet.http.*;
public class GetCookieServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
Cookie[] cookies = request.getCookies();
String username = "";
if (cookies != null) {
for (Cookie c : cookies) {
if (c.getName().equals("username")) {
username = c.getValue();
}
}
}
response.setContentType("text/html");
PrintWriter out = response.getWriter();
if (!username.equals("")) {
out.println("Welcome back, " + username);
} else {
out.println("No user cookie found.");
```

}
}
}

- In the first servlet, a cookie named "username" is created and added to the client response.
- The browser stores this cookie and sends it with subsequent requests.
- The second servlet reads and identifies the user based on the cookie value. This demonstrates how cookies support user identification across different HTTP requests.

OR

7.a)**Explain the directory structure of a servlet-based web application and the role of the web.xml file** **5M**

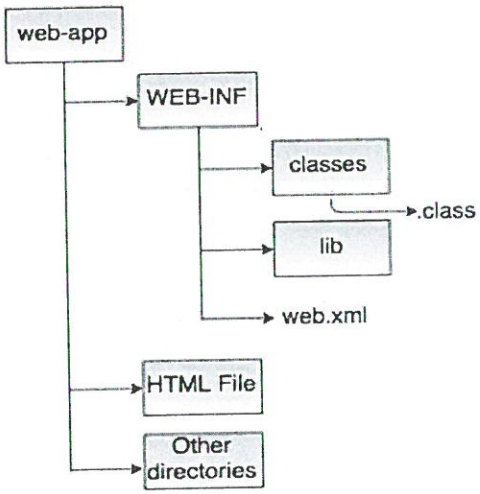


Fig: Directory Structure of Servlet Application

Role of web.xml:

- • Describes servlets, servlet-mapping, filters, listeners, context params, welcome-file list, error pages, security constraints, session config.
- • Example minimal web.xml snippet:

```
<web-app>
<servlet>
<servlet-name>Front</servlet-name>
<servlet-class>com.example.FrontController</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Front</servlet-name>
<url-pattern>/app/*</url-pattern>
</servlet-mapping>
 </web-app>
```

b)How is the HttpServletResponse interface used to send responses? Explain setting headers and redirecting responses. 5M

HttpServletResponse is a servlet interface used to construct and send responses from the server to the client. It allows setting the content type, sending output data, configuring HTTP status codes, adding response headers, and performing redirection. It provides full control over how the browser receives and interprets server-generated data.

Sending Responses Using HttpServletResponse

To send a response, the servlet first specifies the data type and then writes the output.

a) Setting the Content Type

response.setContentType("text/html");

This tells the browser the format of the response (HTML, JSON, PDF, etc.).

b) Writing Output

PrintWriter out = response.getWriter();

out.println("<h1>Welcome User</h1>");

getWriter() is used for text output, whereas getOutputStream() is used for binary data.

Setting HTTP Response Headers

Headers are used to pass additional information to the browser, such as caching rules, download instructions, or metadata.

a) Methods for Setting Headers

- setHeader(String name, String value) – replaces an existing header
- addHeader(String name, String value) – adds multiple values
- setIntHeader(String name, int value) – sets integer header
- setDateHeader(String name, long date) – sets date header

b) Examples of Setting Headers

i) Disabling Cache

response.setHeader("Cache-Control", "no-cache, no-store");

response.setHeader("Pragma", "no-cache");

response.setDateHeader("Expires", 0);

ii) Triggering File Download

response.setHeader("Content-Disposition", "attachment; filename=\"data.pdf\"");

iii) Setting Custom Header

response.setHeader("X-App-Version", "1.0");

Headers help in controlling caching, downloads, refresh timings, and browser behaviour.

Redirecting Responses

Redirection forces the browser to load a new URL. It is commonly used after login, form submission, or permission checks.

a) Redirect Using sendRedirect()

response.sendRedirect("login.jsp");

This method:

1. Sets status code 302 (Found)
2. Sets the Location header with new URL
3. Browser sends a new request to the redirected URL

UNIT-IV

8a)Explain the different phases in the JSP life cycle. Describe the role of methods like jsplnit(), jspServiceQ, and jspDestroy(). 5M

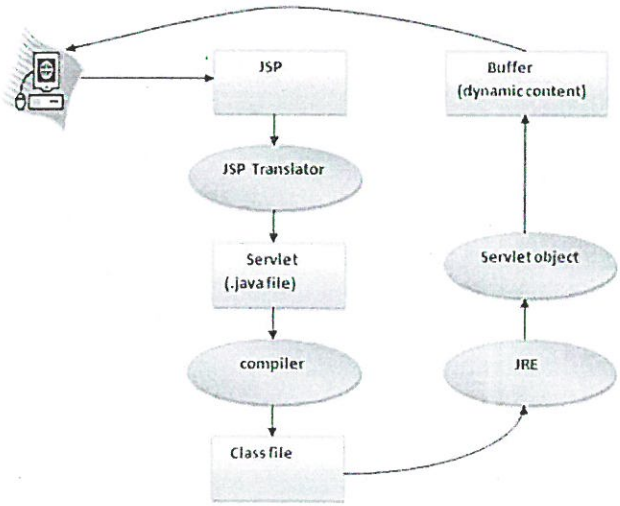
JSPLifeCycle:

TheJSPpagesfollowthesephases:

- TranslationofJSPPageoCompilationofJSPPage
- Classloading(theclassloaderloadsclassfile)

- Instantiation(ObjectoftheGeneratedServletiscreated).
- Initialization(thecontainerinvokesjspInit()method).
- Requestprocessing(thecontainerinvokes_jspService()method).
- Destroy (thecontainerinvokesjspDestroy()method).

Note:jspInit(),_jspService()andjspDestroy()arethelifecyclemethodsofJSP.



Initialization(jspInit())

- ThismethodiscalledonlyoncewhentheJSPisfirstloaded.
- It is used for initializing resources like database connections or configurations.

```
public void jspInit() {
// Initialization code, like setting up resources.
System.out.println("JSP Initialized.");
}
```

RequestProcessing(_jspService())

- Thismethodiscalledforeveryrequest.
- Itcannotbeoverriddenbecauseitisauto-generatedanddeclaredasfinal.
- ItreceivesHttpServletRequestandHttpServletResponseobjectstohandletherequest.

```
publicvoid_jspService(HttpServletRequestrequest,HttpServletResponseresponse){
// Code that handles the request, like generating HTML output.
response.getWriter().write("Hello, World!");
}
```

JSPCleanup(jspDestroy())

- ThismethodiscalledoncebeforeremovingtheJSPfrom service.
- It is used for releasing resources, such as closing database connections or cleaning up open files.

```
publicvoidjspDestroy(){
// Clean up resources like closing database connections.
System.out.println("JSP Destroyed.");
```

}

- This Lifecycle ensures that JSP pages are efficiently compiled, managed and cleaned up by the server container.

b)**Describe the three types of scripting elements in JSP with examples.-5M**

JSPScriptingElements:

InJSP,javacodecanbewritteninsidethejsppageusingthescriptlettag.

JSPScriptingelements

Thescriptingelementsprovidestheability toinsertjavacodeinsidethejsp.There are three types of scripting elements:

- scriptlettag
- expressiontag
- declarationtag

Scriptlets

- A scriptlet is a block of java code that is executed during the request processing time, and is enclosed between <% and %> tags.
- We can embed any amount of java code in the JSP Scriptlets. JSP Engine places these code in the *_jspService()* method.

Example:

```
<html>
</html>
<body>
    <%out.print("WelcometoJSP");%>
</body>
```

JSPexpressiontag:

The most simple and basic of JSP scripting is expression. Expression is used to insert values directly to the output. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

- The syntaxoftheexpressionisasfollows.(benotedthatthereisnospacebetween <%and=)

  <%=expression%>

- For example, if you want to print out the current date and time you can use the expression as follows:

  <%=newjava.util.Date()%>

JSPDeclarationTag:

- The**JSPdeclarationtag**isused*todeclarefieldsandmethods*.
- TheJSPdeclarationissurroundedbythesign**<%!**and**%>**.Forexample,if you want to declare a variable *x,* you can use JSP declaration as follows:
    - <%! intx=10; %>
- Thefinalsemicolonisveryimportant.
- The difference between a variable using declaration and a variable is declaredusingScriptletisthatavariabledeclareusingdeclarationtagis accessiblebyallthemethodswhileavariabledeclaredusingScriptletis only accessible to the method *_jspservice*ofthegeneratedservletfromJSPpage.
- Thesyntaxofthedeclarationtagisasfollows:

<%!fieldormethoddeclaration %>

Example of JSPdeclaration tag that declares field

```
<html>
    <body>
        <%!intdata=50;%>
        <%="Valueofthevariableis:"+data%>
    </body>
</html>
```

Example:Printstheusernamewithexpressiontag.

```
<html>
    <body>
        <%="Welcome"+request.getParameter("uname")%>
    </body>
</html>
```

OR

9a) **Illustratedifferencebetween static include (<°lo@ include %>) and dynamic include (<jsp: include>) with code snippets.**--5M

1. StaticInclude—<%@includefile="..."%>

Howitworks

- Thefile is**included atJSP translationtime**(i.e.,beforetheJSP is compiled).
- Thecontentsoftheincludedfileare**mergedinto theJSP**asiftheywerecopiedand pasted.
- Bestfor**staticcontent**or codethatmustbecompiled together.
- Changestotheincluded file requirea**recompile**ofthe JSP.

Example

main.jsp

```
<html>
<body>
  <%@includefile="header.jsp"%>

  <h2>Welcometothe MainPage</h2>

  <%@include file="footer.jsp" %>
</body>
</html>
```

header.jsp

```
<h1>MyWebsite</h1>
<hr>
```

footer.jsp

```
<hr>
<p>&copy;2025MySite</p>
```

Allcodefromheader.jspandfooter.jsp isinjected**beforecompilation**.

2. DynamicInclude—<jsp:includepage="..."/> How it works

- Thefile is included**atrequesttime**, eachtimethepage is requested.
- Theincludedfileisexecuted separately,andits **output**is insertedintotheparentJSP.
- Bestfor**dynamiccontent**(e.g.,contentthatchangesfrequently).
- Changesappear**immediately**,norecompilation needed.

Example

main.jsp

```
<html>
<body>
  <jsp:includepage="header.jsp"/>
  <h2>Welcometothe MainPage</h2>
```

```
    <jsp:includepage="dynamicTime.jsp"/>
    <jsp:includepage="footer.jsp"/>
  </body>
  </html>
```

dynamicTime.jsp

```
<p>Current time:<%=newjava.util.Date()%></p>
```

Eachrequestdynamicallyinsertstheoutputofdynamic Time.jsp,sothetimeupdateson every page refresh.

b) **Prepare a sample JSP pageusing<c: forEach> and <c: if.> tags from JSTL and explain their purpose. --5M**

<c:forEach>tag: The<c:forEach>isaniterationtagusedforrepeatingthe nested body content for fixed number of times or over the collection.

Syntax:

<c:forEachattributes>body</c:forEach>

Attributes are:

items – specifies the collection to iterate over
var–
specifiesthenameofthevariable(for(inti=0;i<10
;i++)
begin–Startingindexof theloop (0)
end–takestheendingindexfortheloop(10-1=9)
step–Anoptionalincrementfortheloop.Defaultis1.(++)

```
<%@tagliburi="http://java.sun.com/jsp/jstl/core"prefix="c"%>
<html>
    <head>
        <title>CoreTagExample</title>
    </head>

    <body>
        <c:forEachvar="j"begin="1"end="100"
            step="2"> Item
            <c:outvalue="${j}"/><p>
        </c:forEach>
    </body>
</html>
```

<c:if>tag: The<c:if>tagis usedfortestingtheconditionanditdisplaythebody content, if the expression evaluated is true.

Ex:

- Attribute–test–Booleanvariable

```
<%@tagliburi="http://java.sun.com/jsp/jstl/core"prefix="c"%>
<html
        <head>
                <title>CoreTagExample</title>
        </head>
        <body>
                <c:setvar="income"scope="session"value="${4000*4}"/>
                <c:iftest="${income>8000}">
                        <p>Myincomeis:<c:outvalue="${income}"/><p>
                         </c:if>
                   </body>
          </html>
```

SampleJSP Page(Using<c:forEach>and<c:if>)

```
<%@pagecontentType="text/html;charset=UTF-8"language="java"%>
<%@taglibprefix="c"uri="http://java.sun.com/jsp/jstl/core"%>
<html><head>
  <title>JSTLExample</title>
</head>
<body>
<h2>StudentMarks List</h2>
<!--Settingsampledata-->
<c:setvar="marks"value="${[85,60,74,92,48]}"/>
<tableborder="1" cellpadding="8">
  <tr>
    <th>Student No.</th>
    <th>Marks</th>
    <th>Result</th>
  </tr><!--Iteratingthroughmarkslist-->
  <c:forEachvar="m" items="${marks}"varStatus="status">
    <tr>
      <td>${status.count}</td>
      <td>${m}</td>
      <!--Conditionalcheckforpass/fail-->
      <td>
        <c:iftest="${m>=50}">
            Pass
          </c:if>
        <c:iftest="${m<50}">Fail
        </c:if>
      </td>
    </tr>
```

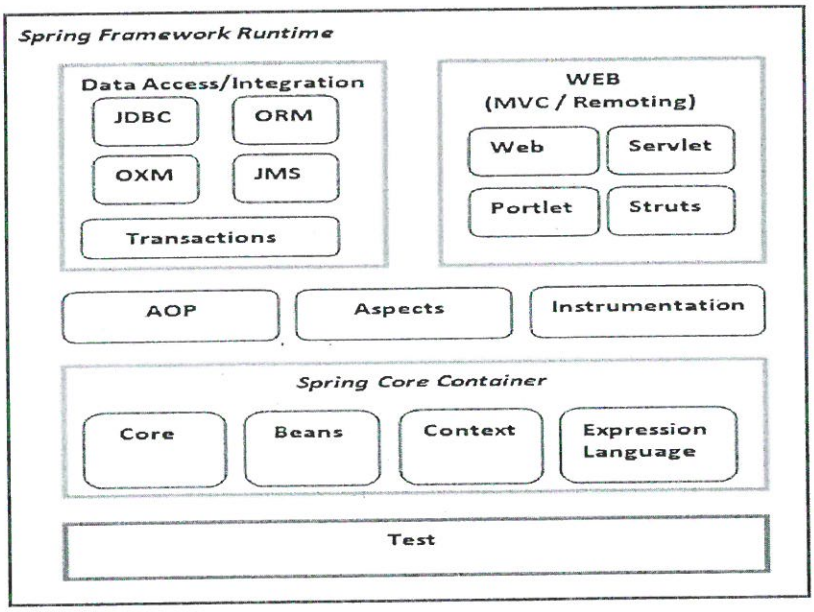
```
    </c:forEach>
</table>

</body>
```

UNIT-V

10. a)**Draw and explain the architecture of theSpring Framework. Briefly describe the role of any four major modules.**5 M

ExploringtheSpringFramework Architecture:

TheSpringFrameworkarchitectureismodularandlayered,builtuponacorecontainerthat provides fundamental functionalities. This modularity allows developers to use only the components necessary for their application, integrating easily with other frameworks.



(a) CoreContainer Layer

This isthefoundation of the entireSpring Framework.
ItprovidestheInversionofControl(IoC)andDependencyInjection(DI) features.
Itmanagesthe creation and wiring(linking)of Java objects -knownasspringBeans.

Main Modules:

- spring-core:Provides coreutilitiesandclasses.
- spring-beans: Managesbeancreation,lifecycle,anddependencyinjection.
- spring-context:TheContextmoduleprovidesawaytoaccessobjectsinaframework- style. Acts like a JNDI registry replacement.
- spring-expression(SpEL):Providesapowerfulexpressionlanguagetoqueryand manipulate bean properties dynamically.

(b) DataAccess/Integration Layer

- Tomakedataaccessand transactionmanagementsimple,consistent, and declarative.

Main Modules:

- spring-jdbc:SimplifiesJDBCcodeandremovesboilerplatecode.
- spring-orm:IntegrateswithORMframeworks (likeHibernate, JPA).
- spring-tx:Managestransactionsdeclaratively(withoutmanual code).
- spring-jms:Provides supportforJMS(Java Messaging Service).
- spring-oxm:Object/XMLmappingsupport.

(c) AOP(Aspect-OrientedProgramming)LayerandAspects.

- Itkeepsourcodecleanbyseparatingcommonfunctionalitiesthatareusedacross multiple layers.

Main Modules:

- spring-aop:ImplementsAOPfeatures.
- spring-aspects:IntegrateswithAspectJ(apopularAOPframework).

Web Layer

- Providestoolsfor creatingweb-basedapplications.
- Supportsbothtraditionalwebapplications(MVCpattern)andmodernRESTfulweb services.

Main Modules:

- spring-web: Basicwebintegrationandfileupload,HTTPhandling,andwebutilities.
- spring-webmvc:ImplementstheModel-View-Controller(MVC)pattern-theheartof Spring Web applications.
- spring-webflux:Supportsreactiveprogrammingforasynchronousandnon-blocking applications.

(d) Instrumentation Layer

- Providesclassinstrumentation andclassloadersupportfor serversandenvironments.
- TheInstrumentationlayerhelpsSpringworksmoothlyinadvancedserverenvironments by allowing it to modify and manage Java classes dynamically during runtime.

(e) TestingLayer

- Provides supportfor unittestingand integrationtestingof Spring components.
- IntegrateswithJUnitandTestNGframeworks.

b)**Illustratethe difference between constructor-based and setter-based dependency injection in Spring with examples.** ~ 5M.

Constructor-BasedDependencyInjection

- Dependenciesareprovidedthrough theclass**constructor**.
- Ensuresthatthedependencyisavailable atobjectcreationtime.
- Promotes**immutability**(dependenciescanbe final).
- Makesunit testingeasier(nopartiallyinitializedobjects).

Example

```
Dependency
public interface MessageService {
   voidsendMessage(Stringmessage);
}

publicclassEmailMessageServiceimplementsMessageService{
   @Override
   public void sendMessage(String message) {
      System.out.println("Emailsent:"+message);
   }
}
Clientusingconstructorinjection
publicclass NotificationManager{
   privatefinalMessageServicemessageService;
   //Constructor-basedDI
   publicNotificationManager(MessageServicemessageService){
      this.messageService = messageService;
   }
   public void notifyUser(String msg) {
      messageService.sendMessage(msg);
   }

}
SpringConfiguration(JavaConfig)
```

```
@Configuration
publicclassAppConfig{

   @Bean
   publicMessageServicemessageService(){ return
      new EmailMessageService();
   }

   @Bean
   publicNotificationManagernotificationManager() {
      returnnewNotificationManager(messageService());
   }
}
```

Setter-BasedDependency Injection

- Dependenciesareprovided through**settermethods**.
- Allowsfor**optional** dependencies.
- More flexible, but canleadto partiallyconstructedobjects.
- Usefulwhen the dependencyshouldbe changeable at runtime.

Client usingsetter injection

```
publicclass NotificationManager{

   privateMessageServicemessageService;

   //Setter-basedDI
   publicvoidsetMessageService(MessageServicemessageService){ this.messageService =
      messageService;
   }

   public void notifyUser(String msg) {
      messageService.sendMessage(msg);
   }
}
```

SpringConfiguration(JavaConfig)

```
@Configuration
publicclassAppConfig{

   @Bean
   publicMessageServicemessageService(){ return
      new EmailMessageService();
   }
```

```
    @Bean
    publicNotificationManagernotificationManager() {
        NotificationManagermanager=newNotificationManager();
        manager.setMessageService(messageService());
        returnmanager;
    }
}
```

OR

11a)**How does setter injectionworkin Spring? Prepare a code example using XML configuration.5**M

How SetterInjectionWorksinSpring

Setter-baseddependencyinjectionmeansSpringinjects dependenciesinto abeanbycalling its **setter methods** after the bean is created.
This allows:

- Optional dependencies
- Changingdependenciesafterobjectcreation
- Moreflexibleconfiguration

Springidentifiesthedependencythroughthesettermethodname(e.g.,setMessageService) and injects the required bean defined in the XML configuration.

Example:SetterInjectionUsingXMLConfiguration

1. ServiceInterfaceandImplementation

```
public interface MessageService {
    voidsendMessage(Stringmessage);
}
publicclassEmailMessageServiceimplementsMessageService{
    @Override
    publicvoidsendMessage(Stringmessage){

        System.out.println("Email: "+ message);
    }
}
```

ClientClassUsingSetterInjection

```
publicclass NotificationManager{
```

```
    privateMessageServicemessageService;

    //SetterforDI
    publicvoidsetMessageService(MessageServicemessageService){ this.messageService =
      messageService;
    }

    public void notifyUser(String msg) {
      messageService.sendMessage(msg);
    }
}
```

XMLConfiguration (applicationContext.xml)

```
<?xmlversion="1.0"encoding="UTF-8"?>
<beans   xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
      http://www.springframework.org/schema/beanshttp://www.springfra
      mework.org/schema/beans/spring-beans.xsd">

  <!--Dependencybean-->
  <beanid="messageService"class="com.example.EmailMessageService"/>

  <!--Client bean usingsetterinjection -->
  <beanid="notificationManager" class="com.example.NotificationManager">
    <propertyname="messageService"ref="messageService"/>
  </bean>

</beans>
```

2. Runningthe Application

```
importorg.springframework.context.ApplicationContext;
importorg.springframework.context.support.ClassPathXmlApplicationContext;

publicclass Main{
  publicstaticvoidmain(String[]args){
    ApplicationContext context =
        newClassPathXmlApplicationContext("applicationContext.xml");

    NotificationManagermanager=(NotificationManager)
context.getBean("notificationManager");

    manager.notifyUser("HelloviaSetterInjection!");
```

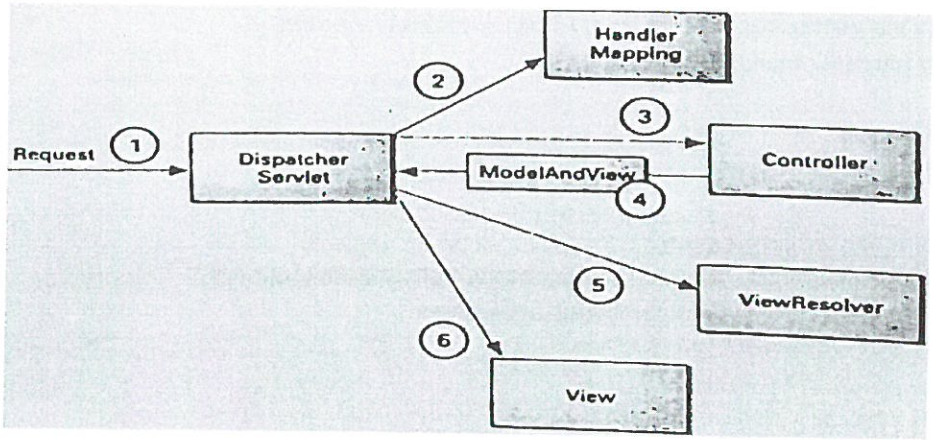
```
    }
}
```

b)**Describe the Spring MVC life cycle with a neat diagram. Explain the role of DispatcherServlet**5 M

SpringMVCRequestProcessingLifeCycle

SpringMVC follows awell-defined flowforeveryincomingHTTP request.

1. **Clientsendsrequest**toaSpringMVCapplication.
2. **DispatcherServlet**receivestherequest(frontcontroller).
3. **HandlerMapping**determinestheappropriatecontroller.
4. **DispatcherServlet**forwardstherequesttothechosencontroller.
5. **Controller**processesthe request andreturnsa**ModelAndView**.
6. **ViewResolver**determines whichviewtorender.
7. **DispatcherServlet**dispatchesthe model to the chosen view.
8. **View**rendersthe response.
9. **Response**is sent backto theclient.



Roleof theDispatcherServlet

- The**DispatcherServlet**is the**heartof SpringMVC.**
- Itactsasthe **FrontController**intheFrontControllerdesign pattern.

KeyResponsibilities

1. **ReceiveAllIncoming Requests**

   Allwebrequests passthroughDispatcherServlet(*usually mappedas*/*or /).

2. **Find theRightController(HandlerMapping)**
   ItaskstheHandlerMappingtodeterminewhichcontroller/methodshouldhandlethe request.
3. **InvoketheController**
   Oncethehandlerisidentified,DispatcherServlet:

   - Invokesthecontrollermethod
   - Passesrequest parameters
   - Receivesthe **ModelAndView**result
4. **DeterminetheView**
   Ituses the ViewResolver to selectthecorrect UI component:
   - JSP
   - Thymeleaf
   - FreeMarker
   - JSON(for @RestController)
5. **Render the Final Response**
   Itgivesthemodeldata tothe chosenviewand sendsthe producedoutput tothe browser.
6. **Orchestrates the Entire MVC Flow**
   DispatcherServletisnotdoingtheactualbusinesslogic—itsimply**coordinates**between the components.

Spring MVC Life Cycle

Client→DispatcherServlet→HandlerMapping→Controller→ViewResolver→View→ Response

DispatcherServlet Role

- FrontControllerfor allMVCrequests
- Finds controllers
- Delegatesto handlers
- Resolves views
- Preparesandsends theresponse